|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ) КАФЕДРА «Информационная безопасность» (ИУ8)

Отчёт

по лабораторной работе № 5

по дисциплине «Интеллектуальные технологии информационной безопасности»

**Тема: «Исследование рекуррентной нейронной сети Хопфилда на примере задачи распознавания образов»**

Вариант 11

Выполнил: Мишков А.О., студент группы ИУ8-62

Проверил: Строганов И.С., преподаватель каф. ИУ8

г. Москва, 2022 г.

# Цель работы

Исследовать процедуры обучения и функционирования рекуррентной нейронной сети (РНС) Хопфилда в качестве устройства автоассоциативной памяти.

# Условие

Задано 3 образа для запоминания: S, T, U

Представим образы в виде биполярных матриц-паттернов размерности 5х3

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 | 1 |
| 1 | -1 | -1 | -1 | 1 | -1 | 1 | -1 | 1 |
| 1 | 1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 |
| -1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 |
| 1 | 1 | 1 | -1 | 1 | -1 | 1 | 1 | 1 |

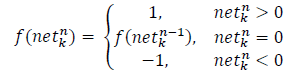
Режим работы РНС Хопфилда: асинхронный

# Теоретическая часть

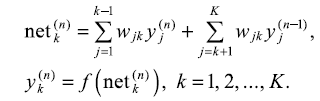
РНС Хопфилда является автоассоциативной памятью, которая в ответ на входное воздействие-сигнал

формирует отклик

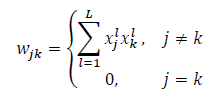
структурно соответствующий прототипу.

Функция активации:

𝑘

В асинхронном режиме каждая эпоха с номером n=1,2,… включает в себя следующие вычисления:

Для начала работы РНС Хопфилда необходимо задать начальные условия:

а также вычислить компоненты матрицы весов:

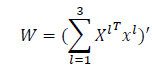
# Ход работы

Вначале необходимо представить образы в виде векторов, которые составлены на основе биполярных матриц-паттернов.

S = [1, 1, 1, -1, 1, 1, -1, 1, -1, 1, 1, -1, 1, 1, 1]

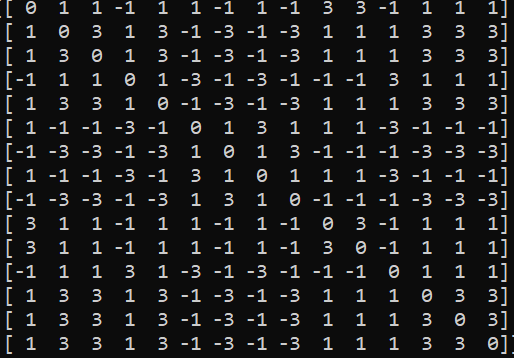
T = [1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1]

U = [1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1]

Следующий шаг – настройка веса РНС Хопфилда

Полученная матрица весов представлена в таблице 1.

Таблица 1 – Матрица весов



Теперь поочередно исказим все образы, а потом восстановим их с помощью

нейронной сети. Результаты восстановления представлены на рисунках 1-3.

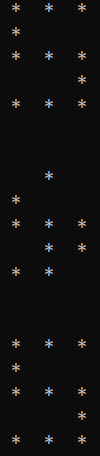


Рисунок 1 – Искажение S



Рисунок 2 – Искажение T

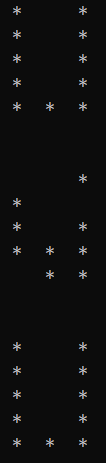


Рисунок 3 – Искажение U

# Выводы

В ходе выполнения данной лабораторной работы было произведено обучение рекуррентной нейронной сети (РНС) Хопфилда в качестве устройства автоассоциативной памяти. Все искаженные образцы были успешно восстановлены.

# Приложение А. Исходный код программы

import numpy as np

class Hopfield:

def \_\_init\_\_(self, size: int):

self.size = size

self.we\_ma = np.zeros((size, size), dtype=int)

def val(self, net: int, f\_net: int):

if net > 0:

return 1

elif net < 0:

return -1

else:

return f\_net

def matr\_ves(self, vec\_standards):

for j in range(self.size):

for k in range(self.size):

if j == k:

self.we\_ma[j][k] = 0

else:

self.we\_ma[j][k] = sum(

[vec\_standards[l][j] \* vec\_standards[l][k] for l in range(len(vec\_standards))])

def rasp(self, vec\_input):

vec\_y = [0 for \_ in range(len(vec\_input))]

while not (np.array\_equal(vec\_y, vec\_input)):

for k in range(len(vec\_input)):

net = 0

for j in range(k):

net += self.we\_ma[j][k] \* vec\_y[j]

for j in range(k + 1, len(vec\_input)):

net += self.we\_ma[j][k] \* vec\_input[j]

vec\_y[k] = self.val(net, vec\_input[k])

vec\_input = vec\_y

return vec\_y

def prin(x):

for i in range(5):

s = ''

if x[i] == -1:

s += ' '

else:

s += ' \* '

if x[i + 5] == -1:

s += ' '

else:

s += ' \* '

if x[i + 10] == -1:

s += ' '

else:

s += ' \* '

print(s)

print('\n')

if \_\_name\_\_ == '\_\_main\_\_':

S = [+1, +1, +1,

-1, +1, +1,

-1, +1, -1,

+1, +1, -1,

+1, +1, +1]

T = [+1, -1, -1,

-1, -1, +1,

+1, +1, +1,

+1, +1, -1,

-1, -1, -1]

U = [+1, +1, +1,

+1, +1, -1,

-1, -1, -1,

+1, +1, +1,

+1, +1, +1]

SS = [-1, +1, +1,

-1, +1, +1,

-1, +1, +1,

+1, -1, -1,

+1, +1, -1]

TT =[-1, +1, -1,

-1, -1, +1,

+1, +1, -1,

+1, +1, +1,

-1, -1, -1]

UU =[-1, +1, +1,

+1, -1, -1,

-1, -1, +1,

+1, +1, -1,

+1, +1, +1]

network = Hopfield(5 \* 3)

network.matr\_ves([S, T, U])

print(network.we\_ma)

prin(S)

prin(SS)

prin(network.rasp(SS))

prin(T)

prin(TT)

prin(network.rasp(TT))

prin(U)

prin(UU)

prin(network.rasp(UU))